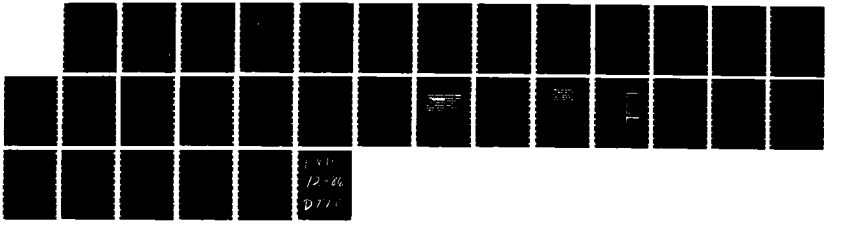
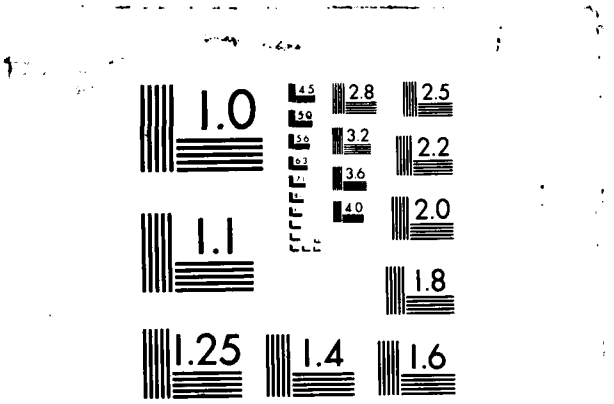
AD-A173 280 THE ISI (INFORMATION SCIENCES INSTITUTE) EXPERIMENTAL MULTIMEDIA MAIL SYSTEM(U) INFORMATION SCIENCES INST MARINA DEL REY CA J B POSTEL ET AL SEP 86 1/1

UNCLASSIFIED ISI/RR-86-173 F/G 9/2 NL





ISI Research Report
ISI/RR-86-173
September 1986

(12)

Jonathan B. Postel
Gregory G. Finn
Alan R. Katz
Joyce K. Reynolds

University of Southern California

The ISI Experimental Multimedia Mail System

INFORMATION SCIENCES INSTITUTE

213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT This document is approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ISI/RR-86-173	5. MONITORING ORGANIZATION REPORT NUMBER(S) ---
6a. NAME OF PERFORMING ORGANIZATION USC/Information Sciences Institute 6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State, and ZIP Code)* 4676 Admiralty Way Marina del Rey, CA 90292	7b. ADDRESS *(City, State, and ZIP Code)*
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Advanced Research Projects Agency 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903 81 C 0335
8c. ADDRESS *(City, State, and ZIP Code)* 1400 Wilson Boulevard Arlington, VA 22209	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. --- / PROJECT NO. --- / TASK NO. --- / WORK UNIT ACCESSION NO. ---

11. TITLE *(Include Security Classification)*
The ISI Experimental Multimedia Mail System (Unclassified)

12. PERSONAL AUTHOR(S) Postel, Jonathan B.; Finn, Gregory G.; Katz, Alan R.; Reynolds, Joyce K.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Research Report	FROM _____ TO _____	1986, September	32

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	ARPA-Internet, bitmap, computer mail, electronic mail, facsimile, mail protocol, message structure, multimedia mail, packet voice
09	02		

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This report discusses a computer-based experimental multimedia mail system that allows the user to read, create, edit, send, and receive messages containing text, images, and voice

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☑ UNCLASSIFIED/UNLIMITED ☑ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Sheila Coyazo Victor Brown	22b. TELEPHONE *(Include Area Code)* 213-822-1511 / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

ISI Research Report
ISI/RR-86-173
September 1986

Jonathan B. Postel
Gregory G. Finn
Alan R. Katz
Joyce K. Reynolds

University of Southern California

The ISI Experimental Multimedia Mail System

INFORMATION SCIENCES INSTITUTE

213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Defense Advanced Research Projects Agency under Contract No MDA903 81 C 0335 Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U S Government, or any person or agency connected with them

ISI Reprint Series

This report is one in a series of reprints of articles and papers written by ISI research staff and published in professional journals and conference proceedings. For a complete list of ISI reports, write to

Document Distribution
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
USA

Introduction

With multimedia computer mail, a user may create messages containing text, image, and voice data and send such messages to other users within a computer network. This paper describes the development, implementation, and use of one such system. The following five sections describe the overview of the system, the system model, the presentation model, the multimedia mail program from the user's point of view, and plans for future work.

1. Overview

The system described here was developed by the University of Southern California (USC) Information Sciences Institute (ISI) as part of the broadly based program of research into computer network communication sponsored by the United States Department of Defense Advanced Research Projects Agency (DARPA). One of the unexpected successes of the computer network experiments of the early 1970's was the nearly universal popularity of computer-based mail. A computer mail system fills the gap that exists between written communication by letter and spoken communication by telephone. The extremely rapid adoption of this new form of communication by those with access to it demonstrated that it filled an important need. The growth of computer networking, the emerging display and software technology, and the decreased cost of computation makes sophisticated forms of communication possible. The DARPA multimedia mail effort is an attempt to employ all these advances to produce a system better suited to the communication of information and ideas.

DARPA has sponsored research on various types of computer networks, including the ARPANET, digital packet radio networks, and digital packet satellite networks. A family of protocols was developed to allow all hosts in the interconnected set of these networks, as well as local area networks developed by others, to communicate with one another (the interconnected set is referred to as the Internet). In

particular, the Internet Protocol (IP) [1,2,3] and the Transmission Control Protocol (TCP)[4,5] were developed.

Host computers in the Internet can range from very small personal computers on local area networks to large mainframes with many users on long-haul networks such as the ARPANET. Data communication between these hosts is built upon IP datagrams. Datagrams provide a simple and flexible basis for internetwork communications. End-to-end reliability is provided by higher level protocols (such as TCP). Networks within the Internet are connected to one another via gateways which route datagrams through the various networks, from gateway to gateway, until they arrive at their destination [7]. In addition to IP and TCP, the User Datagram Protocol (UDP) [8] allows users to send IP datagrams with added multiplexing and error detection capabilities. The File Transfer Protocol (FTP) [9] is used for the transmission of files using TCP.

The multimedia mail system is based on a standard for representing multimedia documents in a machine-independent manner, in order to exchange the documents among machines and document systems of possibly dissimilar architectures. Previous computer mail systems typically transmitted their commands and data between computers in text-only form. The protocols used for such a purpose are not designed with the transmission of non-textual data in mind. One of the first tasks in developing a multimedia mail system was the definition and implementation of a mail protocol suitable for the exchange of data in several forms.

The Internet Multimedia Mail Transfer Protocol (MMTP) defines the procedure and format for the transfer of multimedia messages through the Internet [9,10,11]. These messages may contain very general types of data, including text, voice, image, facsimile, and graphics. The protocol is general enough to allow other types of data to be added in the future. The protocol is implemented in a process called a Message Processing Module (MPM). The MPMs are responsible for the routing, transmission, and delivery of messages.

Because differing types of data are transported between machines, each piece of data must be separable and identifiable. A commonly understood mechanism for encoding data of differing types was developed as part of the MMTP protocol. Data elements for short and long integers, floating point numbers, text strings, and

unformatted bit strings are defined. Also included are types designed for collecting related data elements: lists and property-lists. The MPMs exchange data structures called mail bags. Each mail bag is a list of messages. Each message is in turn a property-list of three elements: a transaction identifier, which uniquely identifies the message, a command part, and the documents. The command part of the message contains information used by the MPM to route the message. Some of this information is supplied by a user interface program. The document contains the actual message content, and its interior is neither examined nor defined by the MMTP.

Multimedia messages are created by a user with a User Interface Program (UIP). Messages so created are then submitted to an MPM for delivery. The MMTP assumes that the MPM has available a reliable method of data transmission. In principle, it could use any reliable data stream; in practice, TCP in the ARPA-Internet is used. The definition of the MMTP was largely completed by late 1980. Its first implementation was running on TOPS-20s in the ARPA Internet at approximately the same time. Subsequent implementations have been written by others to run on SUN workstations [12].

The Internet Multimedia Mail Content Protocol (MMCP) defines a format for the document [13,14,15]. The structured format is built on the basic data elements used in the MMTP. The MMCP provides for the composition of complex data objects and their encoding in machine-independent data elements. Each document contains a header and a body. The header portion of a document corresponds to the date, to, from, etc., fields of a typical interoffice memo. The body of the document may be a simple character string or a complex structure of lists or property lists. It can be encrypted in part or in whole. Messages may include data objects that represent text, drawings or images, digitized voice, or spread sheets.

One possible presentation of information in the message body can be compared to a briefing, where the speaker displays slides and other visual aids while providing a verbal commentary. The time coordination of such a presentation is captured in the body structure. Three types of time-ordered presentation control are possible within the document: sequential, simultaneous, and independent. Sequential data is presented one frame at a time, in the order listed. Simultaneous data is intended for synchronous viewing, and independent data can be presented in any time order.

For example, one may have text displayed alone, then have graphics displayed while simultaneously listening to a voice description of the graphic drawing.

The DARPA research program in multimedia mail is an experiment to learn about the future direction of computer mail. The primary goal is to provide specifications for computer-oriented data structures to communicate various types of data in messages, including text, graphics, and voice. Demonstrations of the multimedia mail system and experiments with user interfaces are being conducted. This experience has reinforced our belief that more powerful primitives require a high level of abstraction with tightly constrained formats and predefined data types to make rapid progress in complex applications.

2. The System Model

The multimedia mail system is an application in the DARPA Internet Protocol Architecture (Figure 1) [1,16,17,18]. The system model has two major components: a Message Processing Module (MPM), and a User Interface Program (UIP) (Figure 2).

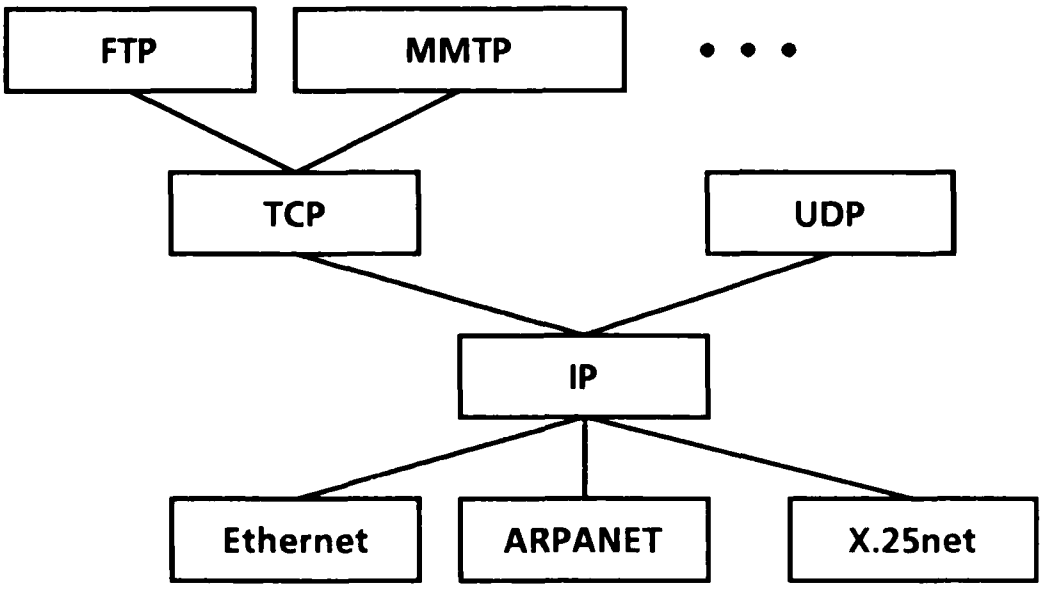


Figure 1: Protocol Architecture

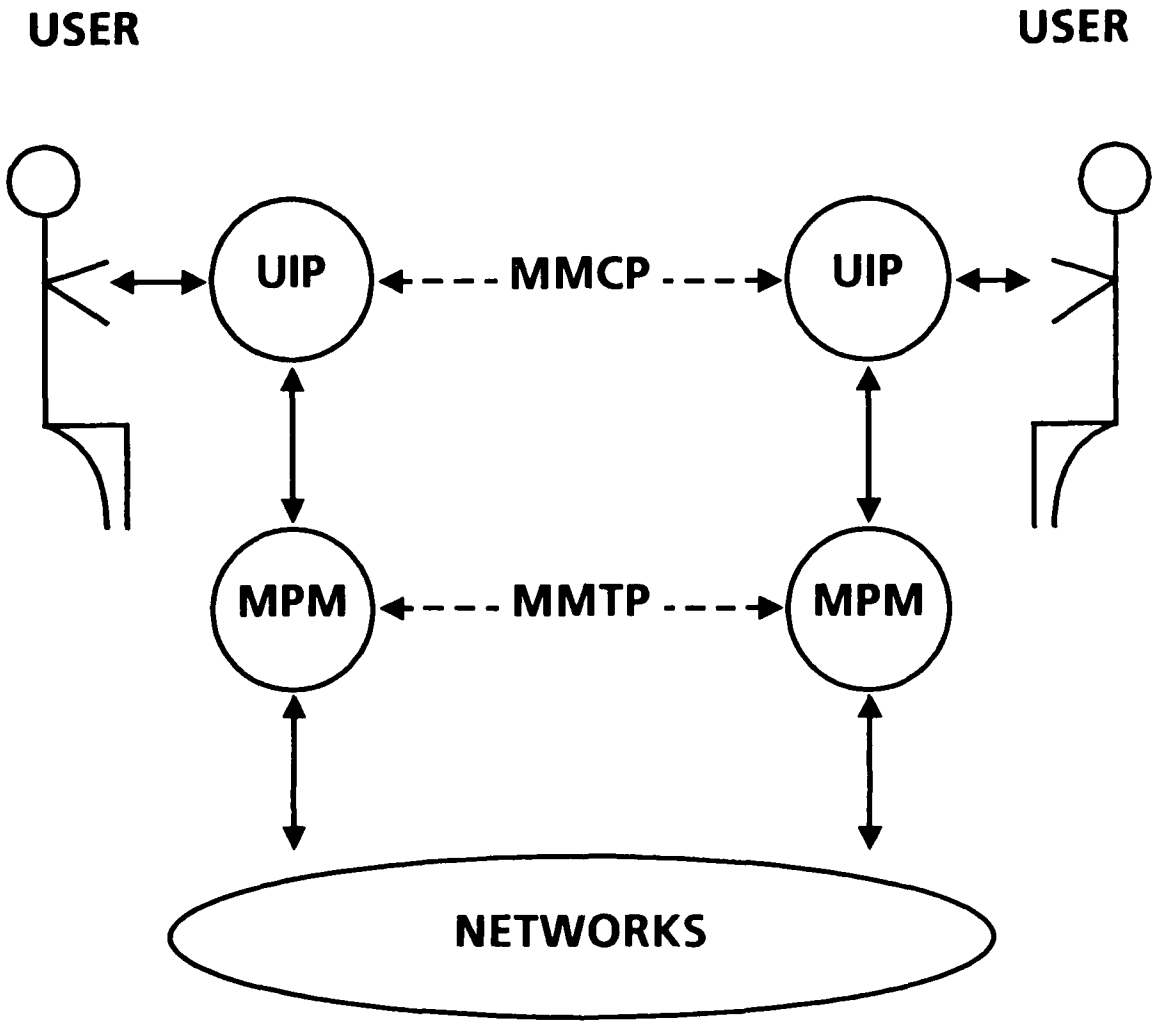


Figure 2: The System Model

The MPM implements the procedures to move mail from one place to another in the communication environment. The UIP interacts with the human user in reading, composing, and sending messages. The development of these protocols has been a joint effort, its initial definition undertaken by ISI, with valuable contributions provided by other institutions as part of the continuing research. It is assumed that many types of UIPs will be implemented and used to compose and display messages delivered by MPM processes.

The user may create a message using various commands of the UIP to prepare various fields of the message and may invoke an editor program to correct or format some or all of the message. Once the user is satisfied with the message, he sends it by placing it in a data structure shared with the MPM.

The MPM accepts the message, examines it, and determines the destination. The destination may be another user on the same host, or a user on another host on the ARPA-Internet. The MPM then adds control information to the message and transmits it. Because the MPMs are responsible for the routing and delivery of messages, it is expected that most organizations or sites that use multimedia mail will operate an MPM.

The MPMs are connected via logical channels. They may communicate control information between themselves by means of commands, and may also interface with local message systems. Since the local message system may be very different from the multimedia mail system, special programs may be necessary to convert incoming multimedia messages to the local format. Likewise, outgoing messages may be converted to the multimedia format.

Because the MPM needs a reliable communication procedure to communicate with other MPMs, it calls on a transport-level protocol such as the Transmission Control Protocol (TCP). The interface to such a procedure conventionally provides calls to open and close connections and to send and receive data on a connection, as well as some means to signal and be notified of special conditions (for example, interrupts). The MPM receives input and produces output through data structures that are produced and consumed by UIPs or other programs.

When a destination MPM accepts a message for delivery to the destination user, it sends back an MPM-level acknowledgment to the originating MPM. In most cases, the destination MPM will be on a full-service host and will use some local procedure to deliver the message to the UIP. In a few cases, the destination MPM will be on a restricted-service host and will be an end-point MPM.

Delivery of messages to an end-point MPM (such as in a workstation) may be on a polling basis, using the forward command. The forward command is sent from an end-point MPM to a relay MPM which is known to hold messages for the end-point. Messages are held when the end-point is not in communication with other MPMs.

The relay responds with a forwarding command to the end-point MPM, telling it how many (if any) messages it has held for the end-point. The relay soon sends those held messages to the end-point MPM. In the current implementation, the MPM is a server process listening for a TCP connection on a particular port. This is not intended to limit the implementation choices; other forms of interprocess communication are allowed, and other types of physical interconnection are permitted.

Because the functional requirements of the UIP for manipulating image data (and other complex data objects) are well matched to the capabilities of some professional workstations, the UIPs are typically implemented on such workstations (for example, SUN and Xerox 1108). Continuous service and reliable file storage however, are better supplied by large service hosts. Therefore, the MPMs are typically implemented on such hosts (for example, TOPS-20, and large UNIX servers).

The interface between the MPM and the UIP is left as a local implementation decision. One possibility is for the MPM to receive input and produce output through data structures produced and consumed (respectively) by the UIP. Since the UIP is typically implemented on a workstation host, and the MPM is implemented on a service host, communication between them may be simply via files deposited by one program and picked up by the other. This may be accomplished via one of the Internet standard file transfer protocols (FTP, TFTP) [8,19]. The format of the files passed between the UIP and the MPM is up to each UIP/MPM interface pair. One reasonable choice is to use the same data structures used in the MPM-MPM communication.

3. Presentation Model

A multimedia message may contain several streams of data. The message composer must have sufficient control to coordinate the display of theses streams in order to effectively communicate his ideas to the recipient. This is accomplished by incorporating explicit sequencing controls into the body of the message. Also important is the ability to control where data is to be placed when it is displayed.

3.1. Sequence Control

In a text-only system, message content is displayed serially from start to finish. Since there is only one medium, it makes no sense to consider parallel presentation of data streams. In multimedia mail, however, there may be several streams which together make up the message. For example, a voice-over may accompany a picture, or text may annotate a map. It would disrupt a message and make it more difficult to understand if the voice-over preceded the picture, or if the annotations appeared before the map image. A mechanism is needed to allow the message composer to control the sequencing of the message parts during the presentation.

Sequence control is accomplished by constructing a multimedia message from a sequence of presentation descriptors. Each presentation descriptor is a property-list containing a sequencing property name. The property value is a list of presentation elements which contain the media-specific data and display information, or, for complex messages, a list of further presentation descriptors.

When a presentation descriptor is *independent,* the composer is indicating that the presentation order of the presentation elements contained in the list is not important. If voice-over accompanies a picture, the picture and voice data presentation elements can be grouped under the *simultaneous* property. When the text must appear after an image, the *sequential* property is used.

3.2. Data Placement

Three distinctly different UIP models for the display of data have been examined through experimental implementations. In the first-generation UIP, a specific area of the screen was reserved for each different medium (voice, image, and text) [12,20,21]. This was found to be cumbersome for several reasons. First, not all the media are present in all messages; that often wasted screen area that could otherwise have been utilized. Second, such reservations restrict the size of a particular media area; if the image is larger than the area reserved for it, the image is clipped. Third, the number of media is not fixed at three; graphics, facsimile, and spreadsheet are examples of other media which have been implemented in some

UIPs. There is clearly a point beyond which it makes no sense to subdivide the screen. A more sophisticated model was needed.

The second model assumes that a message is displayed on a virtual scroll of indefinite width and length. Presentation elements are displayed one after another down the scroll. The reader scans a message by moving a window through the scroll. There is no concept of a page in this model.

The third approach is to display the message data within a conceptual page mapped to a message window on the screen. This message window is an area of the screen which represents an A4 sheet of paper [22]. The message window's origin is its upper left-hand corner. It is intended that all presentation element data displayed be confined to this conceptual page. When constructing a presentation element, the message composer confines himself to an area within the message window.

The information which specifies data placement accompanies each presentation element. This *spatial* control information describes an area within the message window relative to its origin [23,24]. The composer invokes a window-oriented editor for each media type (a text editor for text, a bitmap editor for images, etc.). The window creation and shaping tools provided by each editor define an area used when editing. The *spatial* control information is extracted from the window system when editing is complete; the composer does not explicitly specify it.

3.3. The Media

Three core media are implemented by UIPs. They are *text*, *image*, and *voice*. Each is discussed in some detail below. The remaining media are not implemented by all of the participating institutions. This is primarily due to the lack of acceptable data representation standards for those media when the experiment began.

3.3.1. Text

Text is the medium most commonly associated with computer mail and is the easiest to implement. Text is grouped into lists which share a similar manner of presentation. Each text string within a list contains a separate paragraph.

Paragraphs are separated by at least one blank line when displayed. There are four types of text presentation: *paragraph*, *itemize*, *enumerate*, and *verbatim*.

It is assumed that an implementation has a default font, for example, a 10-point fixed-width font. *Caption*, *font*, *size*, and *style* directives are available as suggestions and may be ignored. There is no assumption that consistent families of fonts are available in all implementations. Because differences in presentation may make it impossible to include all of the text within the suggested *spatial* display area, it is assumed that the implementation can scroll text into and out of the display area.

The verbatim presentation requires the text to be presented without any receiver-supplied punctuation. No filling, justification, or line breaks are inserted by the receiving implementation. The three remaining protocols assume automatic filling, with justification at the receiving implementation's discretion. For the *itemize* and *enumerate* presentations, the first line of each paragraph is preceded by a character or numeral respectively. The text is presented with a hanging indention.

3.3.2. Image

Current implementations manipulate one-bit-per-pixel images. This is due primarily to the nature of the workstation displays. The *bitmap* protocol is used to encode these images [25]. Primitives supplied with workstations usually make this protocol straightforward to implement. The composer has available a number of tools to create, shape, and edit pictures. These tools include the capability of grabbing part of the screen, reading pictures from files, freehand editing, and so on. Although defined, the *greyscale* and *RGB* protocols have not been used due to hardware restrictions [26]. As before, if the image cannot fit within the *spatial* display area suggested, it is assumed that the reader can scroll an image in the display area. Most sites also have facilities for encoding and decoding bitmap data into and out of the facsimile protocol.

3.3.3. Voice

The current voice data standard specifies a format for digitized compressed coding of the voice data using a linear predictive coding algorithm. Other digitized codings

for voice are provided for, including PCM, but have not been used. One issue is the number of data bits needed to represent even a short utterance. The algorithm used requires about 2400 bits per second of speech, and is near the limit of compression while retaining reasonable quality [27].

3.3.4. Other Media

The other media, *facsimile*, *graphics*, and *spreadsheet*, are not implemented by all UIPs. With the exception of *facsimile*, the representation standards for them are not available.

The representations currently used for both *spreadsheet* and *graphics* were defined by Bolt Beranek and Newman (BBN) [28,29,30]. It is understood that this is an interim solution. We await the acceptance of a graphics encoding and transmission standard. For graphics, the GKS metafile or its eventual derivative is the most likely candidate. The spreadsheet medium is an interesting addition. It differs from the other media because it sends along with the data table the data modification rule model. The reader can manipulate the spreadsheet data received and.can examine *alternatives.*

5. The ISI Multimedia Mail Program

At ISI, our current UIP program is the Multimedia Mail Handler (MMH). This is our second version of a UIP. The MMH is written in Interlisp and run on Xerox 1108 machines. The Xerox workstations are connected to a 10-Mbit Ethernet. The MPM at ISI is written in BLISS and Macro 10, and runs on the TOPS-20 system on PDP-10 machines. The PDP-10 is connected to the ARPANET. The ARPANET and the Ethernet are connected via a Gateway. All communication between the various machines in this system uses the Internet protocols. The first UIP, called MMM, was written in Pascal for the Perq workstation computer [20,31]. The Perq workstations are connected to a 3-Mbit Experimental Ethernet. During the initial testing of the MMH, messages were exchanged between the MMM and The MMH (Figure 3).

ISI currently uses a Rapicom-450 facsimile machine for scanning images from offline sources to create facsimile data files or bitmap data files. A facsimile page (8 1/2 by 11 inches) converted to a bitmap is 1726 by 2200 bits (at 200 bits per inch) [26,32,33].

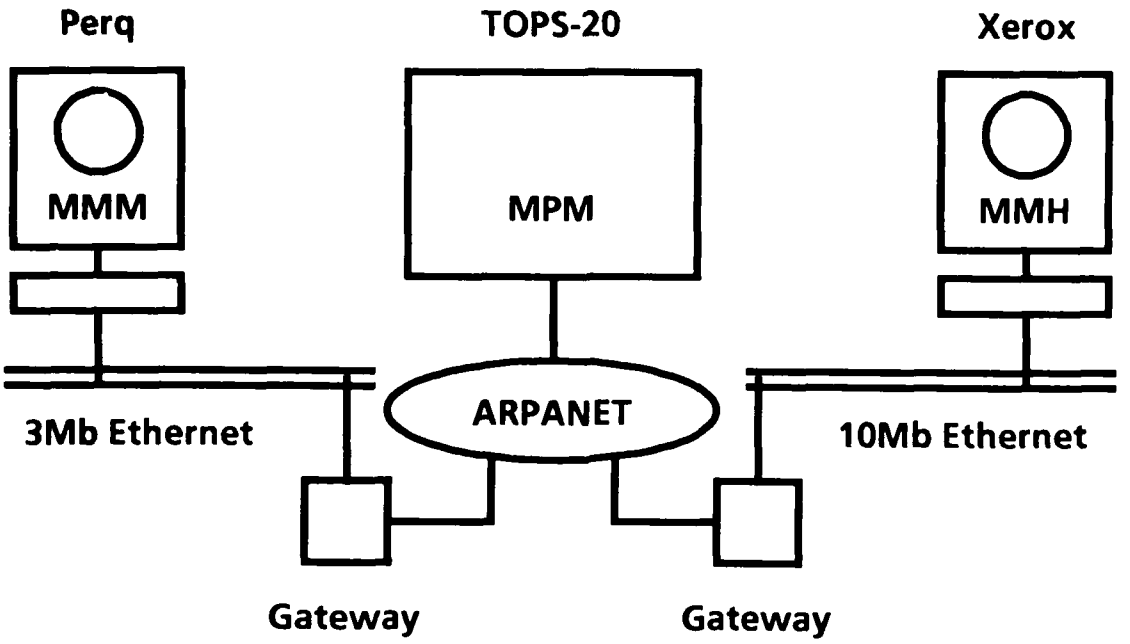


Figure 3: The ISI Configuration

The Rapicom-450 has three picture quality modes. In *fine detail* mode, all of the document is encoded. In *quality* mode, only every other scan line is used, and it is intended that these missing lines are filled in on playback by replicating the previous line. In *express* mode, only every third line is used. Data is encoded two lines at a time, using a special two-dimensional run-length encoding scheme. The facsimile machine is connected to an ISI-developed, microprocessor-based unit called Faxie. Faxie is connected to a TOPS-20 terminal line, and a program is used to read data over this line and store it in a file. The decoding program reads the facsimile data from this file.

A set of programs (FAXTRN, FAXSRV, and CNVFAX), which convert between Rapicom format facsimile files, CCITT format facsimile files, and Bitmap format files, have been implemented to enable images to be entered into the system via the facsimile machine, or printed on that machine from files stored on the system [34,35]. ISI is upgrading this capability with new scanners and printers that follow the CCITT T.4 standards [36]. Programs are available to manipulate bitmap images on the Xerox 1108.

Voice is digitized data in a 2400 baud linear predictive coding format (LPCM) . A central voice server, developed at ISI, uses a standard office telephone to play and record voice. ISI's voice server includes a Lincoln Laboratory Packet Voice Terminal (PVT), an Adams-Russell Speech Processing Peripheral (SPP), and the ISI Switched Telephone Network Interface (STNI) [37,38,39,40].

The MMH user interface gives the user control over the command environment and message presentation. When a user first starts MMH, a message information window is displayed. The mail headers (each containing a brief summary of a message) are automatically displayed in the message information window. Whenever a new message is received or created, a new header for that message is displayed. The user initiates a command by selecting it with the mouse. When the *right* mouse button is clicked, the main message command menu is displayed, including: *Header View*, *Auxiliary Commands*, *Check for New Mail*, *Create*, *Expunge*, *Send Queued Mail*, and *Quit*. When the *left* mouse button is clicked, another message command menu is displayed, including: *Answer*, *Delete*, *Display*, *Forward*, *Move*, and *Undelete*.

To show a message, the user invokes the *Display* command with the mouse while it is over the header of the message to be shown. The "To", "From", and "Subject" fields of the message are displayed in another window. A diagram of the message is displayed in a message structure window. After the user has seen this information, the user points to an entity within the message structure window and clicks the *left* mouse button, which causes a menu of two commands to appear: the Display of the entire body of the message, or the Display of the Subtree selected. The user may look at a particular entity within the message, ignoring the order in which the message was originally created, by using the Subtree command. This allows the user to view any selected entity and to edit or store it in a file. The media data is displayed in a window of the appropriate type within the context of the message window (the virtual A4 page). Each media window is positioned and sized according to the *spatial* parameter. If any of the windows is too large, the user may change the size or position of a window by using the standard window management facilities.

The *Create* command is used to construct a message. The user may enter the names and addresses of recipients. The user also enters a subject field, text, bitmaps, and

voice data. Text can be generated by using keyboard input (with itemize, verbatim, enumerate, and paragraph options) or by creating files with a text editor. Bitmap pictures may be created using the bitmap sketching program, or pictures may be captured by scanning printed images on a facsimile machine.

For each media type there are facilities for the preparation and display of that type of data. For text there is a text editor with capabilities and features similar to those commonly found in editors on workstations. For bitmaps there are facilities for saving and retrieving bitmaps from local files, and for clipping and positioning bitmaps on the virtual page, and there is a bitmap editor for creating new bitmaps. The style of interaction with the MMH command level and with each of these media facilities is quite similar. To give the flavor of these interactions, we describe in some detail the facilities for manipulating voice data.

The voice facilities in MMH allow the user to record, play, display, and edit voice. Three types of display windows are used for voice data. The "Voice Editor" is used to create, record, play, display, and edit voice data. There can be more than one Voice Editor window. The "Voice Display" shows already existing voice data, either from a received message or after it has been created with the Voice Editor. The "Voice Buffer" is used in conjunction with the Voice Editor as an intermediate place to store segments of speech.

In all three of these windows, voice data is portrayed as an energy waveform. Displayed voice can be played from any voice window by clicking the *right* and *left* buttons together, in the window. While voice is being played, the corresponding segment in the display will be highlighted. The command menu for a window is always obtained by clicking the *left* mouse button while pointing within the title bar, and the standard window management menu is available when the user clicks the *right* mouse button within the title bar. All windows may be scrolled by means of a scroll bar on the left side.

MMH communicates with the PVT over the Internet using the NVP-II and ST Internet stream-oriented protocols [27,41] (Figure 4). Voice packets are sent from the Xerox 1108 over the 10-Mbit Ethernet, and through the ISI Internet Gateway, the ARPANET, an ST Gateway (which understands the ST protocol), and finally over a LEXNET (a 1-Mbit broadcast network) to the PVT. The PVT makes a connection over ISI's phone system through the STNI, which dials the user's phone number. The SPP

then converts the LPCM data from voice packets into voice, which is played over the phone. To record voice, the user simply speaks into the phone; the SPP converts this into LPCM data which the PVT then sends to the Xerox 1108 in packets using the reverse path.

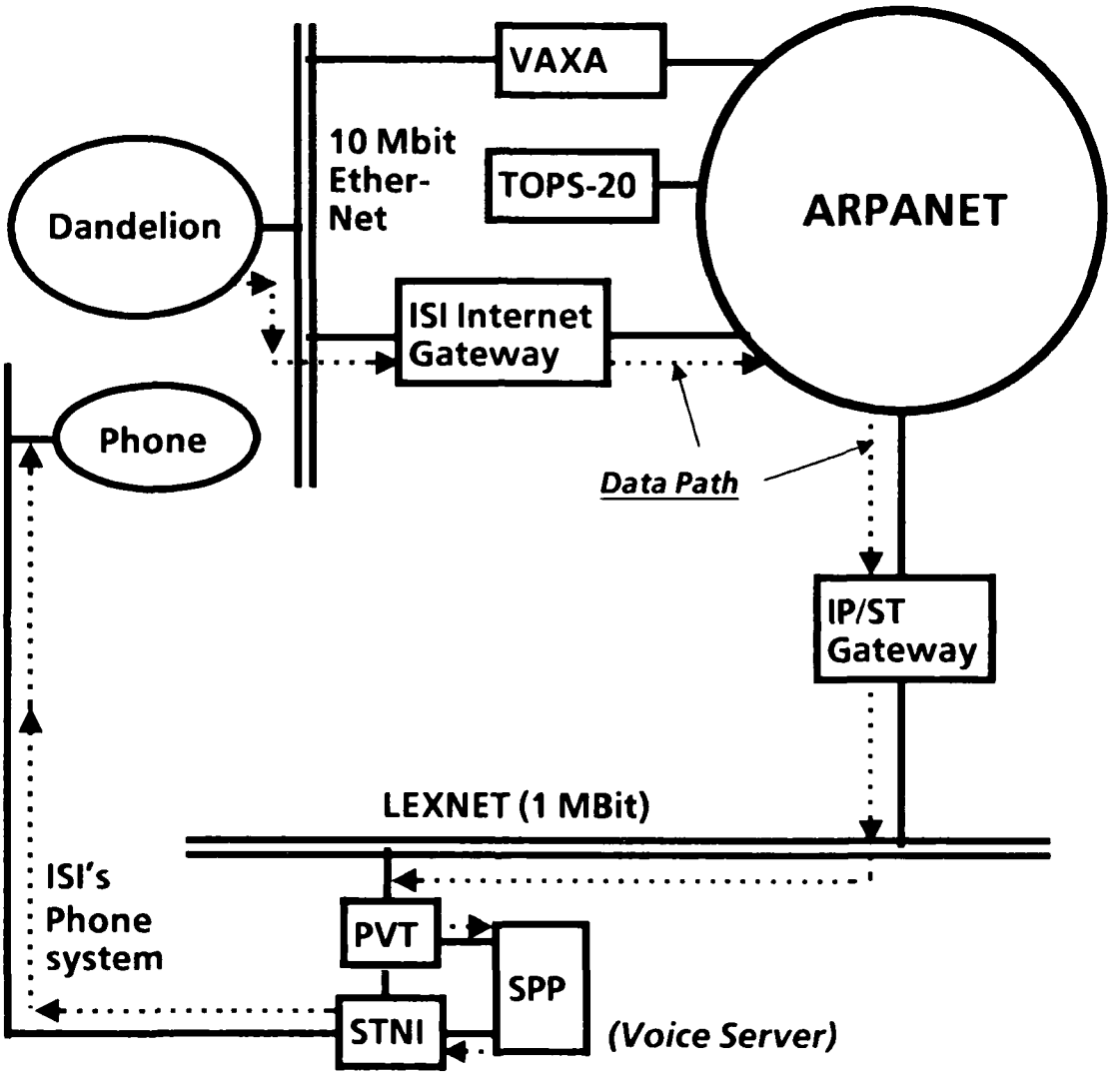


Figure 4: PVT Voice Path and Server Connection

Whenever voice data is to be played or recorded, a connection is set up to the voice server. Since others may want to use the server, a background process running in the workstation will automatically disconnect from it if voice is not played or recorded for about a minute. This background process does not require any action

from the user. If an automatic disconnect occurs, the user will be notified. Any part of the MMH voice system which needs to record or play voice will automatically set up the connection if necessary.

The Voice Editor is entered via the create command in MMH . A standard sized and positioned Voice Editor window is initially created (Figure 5). The user may change

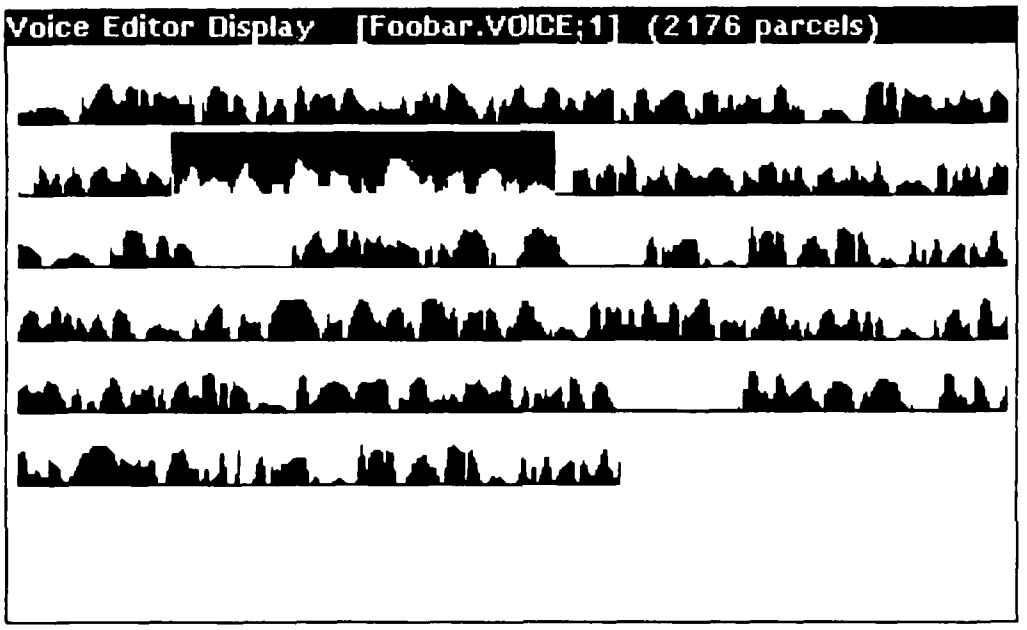


Figure 5: Voice Editor Display and Commands

the size or position of this window at any time with standard window management facilities. When MMH sends the message, the position, size, and shape of the voice window will be included in the *spatial* data of the message. The title bar of the Voice Editor window will show the name of the voice file being edited (if any) and the length of the voice data.

Commands may be selected by pointing in the window's title bar. The *left* mouse button will bring up the Voice Editor command menu, and the *right* button will bring up the standard window management commands (allowing reshaping, moving, and management of the window). Within the Voice Editor window itself (but not in the title bar), voice data is displayed and segments selected. The *left* button designates the left edge of the voice segment selected, and the *right* button

extends the selection. The currently selected voice segment is always highlighted. The *middle* button (or holding down both *left* and *right*) will play the selected segment, setting up the PVT connection if necessary.

The user can read an existing voice file into the display or can record voice directly over the phone via the PVT voice server. Any recorded voice is appended to the end of the existing data. Currently displayed voice can be written into a new file at any time. Editing commands allow the user to delete segments of voice, to copy segments to the Voice Buffer (see below), and to insert either data from the Voice Buffer or silence at any point. Since any Voice Editor window may access the Voice Buffer, data may be transferred from one Voice Editor window to another via the Voice Buffer. Additional commands allow the user to make a selected segment of voice data louder or to set the pitch parameter to be a constant. A special PVT command menu allows the user to directly set up and terminate connections to the PVT, but this generally will not be used except in debugging.

One may wonder how easy it is to edit voice data from a continuous energy waveform display. It turns out to be quite easy, even though there is not much of an intuitive relationship between the display and what the voice sounds like. One of the reasons for this is the ease of playing and replaying a selected segment of voice. All the user has to do is click the *middle* mouse button to listen. Thus, by selecting various segments and playing them, the user can rapidly find his place in the voice stream. We have also found that users rapidly learn to associate the visual pattern of a particular voice segment with what the voice sounds like once it has been played.

Generally, but not always, spoken words are separated by a small amount of silence. This allows the user to estimate where words begin and end. Then, by playing the selected segment, the user can verify that it is the one he wants or else reselect a new segment.

Whenever existing voice data is played or displayed in the MMH system, it is displayed in the Voice Display window (depicted in Figure 6). This includes voice that was created or edited with the Voice Editor, or voice data from a received message. Clicking the *middle* mouse button in the window, or selecting the Play command, will play all of the voice data in the window. It is also possible to write out the displayed voice to a file.

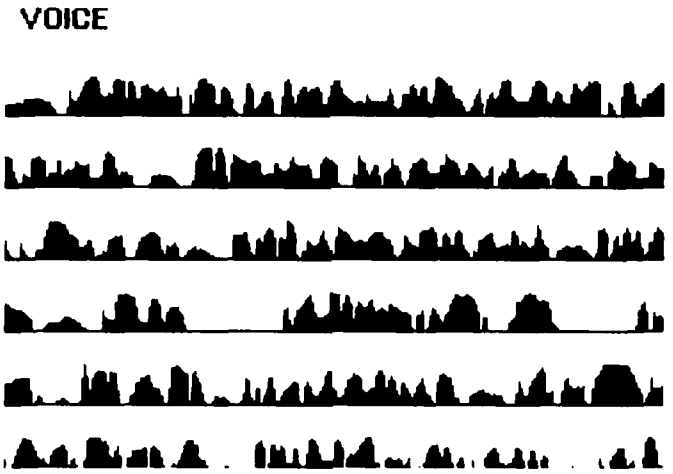


Figure 6: The Voice Display

The Voice Buffer is created the first time the user does a *Segment to Buffer* command. The window will always display what is in the buffer. It may be moved to any position and shaped to any size. The size of the buffer is displayed in the window's title bar. This data may be copied back into a Voice Editor window using the *Insert Buffer* command. There is only one Voice Buffer window.

The user may play the voice data in the buffer by clicking the *middle* mouse button in the window. There are no special commands for the Voice Buffer; however, the standard window management commands are available. The Voice Buffer window is automatically closed when the user quits editing.

Figure 7 shows the screen a user would see when constructing a message. In the lower right is the message structure window. It is easy to see that this message is composed of five parts: three parts text, one image, and one voice. The presentation order is *sequential*. The message window at the left shows the view after the presentation elements have been created. The composer may move or reshape each presentation element window freely until the message is closed. The outlines around the presentation elements are a positioning aid for the composer;

they are absent for the reader. The *spatial* data, which is automatically associated with each presentation element, allows the reader to closely recreate the display shown.

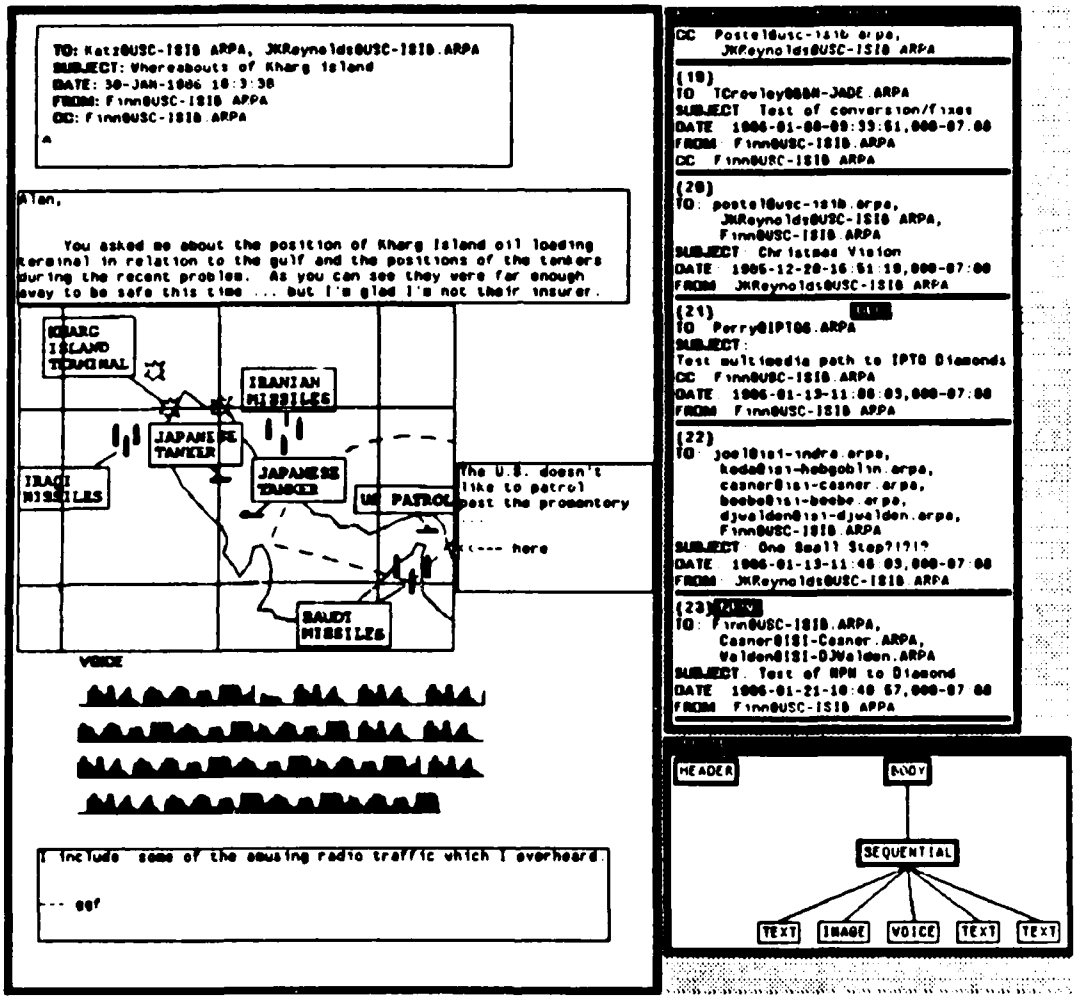


Figure 7: Constructing a Message

Once the user is satisfied with the multimedia message composition, he selects the *Finish* command. The entities are then linked into an internal message structure representation, and MMH returns to main command mode. To send the newly

created message, the user points to the *Send Queued Mail* command and the MMH program delivers the message.

The MMH program is used on a regular basis between workstations. Tests of the multimedia mail capability are performed regularly and demonstrations are frequently conducted for interested visitors. Multimedia mail is also being exchanged on a regular basis by ISI with DARPA-ISTO, BBN, M/A-COM Linkabit, and SRI International [12,21,42].

6. The Future

There are (at least) two points of view about the model of a multimedia message. On one hand are those who feel multimedia messages are a special case of a general document presentation system. In this view, the user goes about creating a message by using typesetting software which converts its output into some standard document interchange format. Such a standard would minimally need control over position, font, and pagination. It must also have graphics and picture capabilities.

Others hold that multimedia messages are too different from documents or books to be included under such a standard without unduly restricting their capabilities. This view stresses media such as voice, which by its very nature cannot be handled by document software, or spreadsheets, which are interactive. While document standards do provide pagination control, they do not include the presentation time ordering thought to be important in multimedia messages.

What has resulted, so far, is an uncomfortable middle ground. Some simpler formatting controls familiar to document preparation systems are included. Examples are data types for automatic filling, enumeration, or itemization, as well as verbatim text presentation. Guidelines for positioning and font choice are also available. Both groups are partly correct. Further pieces from documentation systems need to be incorporated, while retaining those features unique to media like voice and spreadsheets.

Perhaps the most obvious omission in the MMCP is control of paging. This is not an issue to those who view a multimedia message as a single long scroll. It is an issue to composers, who control the positioning of pieces of a message within a frame and

expect similar controls to be available to the reader. In particular, when composing long messages during which the screen frame becomes full, the composer needs to be able to indicate to the reader when the data on the screen should be removed in preparation for what follows (i.e., when the page should be turned).

The potential for multimedia communication in a computer-assisted environment is great. The ability to communicate diagrams or maps and to then talk about them increases the effectiveness of remote communication tremendously. The combination of text, speech, and images into a common framework and data structure may have substantial impact on other applications as well. Computer mail is the most significant use of the new communication capability provided by packet-switching networks. The power of a communication program is directly related to the number of potential communicants. For computer mail, this means that the power of a program is related to the number of people who have access to that program. To have access to a computer mail program requires the use of compatible components: terminals, programs, and protocols. Our work on protocols and programs will increase the power of computer mail by enlarging the set of compatible components. The Multimedia Mail Program is an ongoing experiment to learn about the future direction of computer communication. The primary goal is to provide specifications for computer-oriented data structures to communicate various types of data in messages, including text, images, and voice.

Another aspect of the future of this project is to take into account the advances in international standards in related areas. International standards for computer or electronic mail did not exist when this research was started; today, these standards are defined and are just beginning to be implemented. One task for the future is to reimplement the advanced features of this multimedia mail system on the new base provided by the emerging international standards.

Appendix -- UIP Document Syntax

Notation

This BNF grammar specification presents the production rules for constructing the document body of multimedia messages. The three classes of symbols that may appear in the production rules are terminals, nonterminals, and metasymbols. Metasymbols are used in the grammar as operators. A terminal of the grammar is a symbol that appears literally in the protocols. A nonterminal is a symbol defined to be equivalent to either a sequence of one or more symbols, or any of a set of alternate sequences. The following conventions are used:

(a) Nonterminals are delimited by angle brackets "< >". Thus, <y> means that y is a nonterminal in the grammar.

(b) ":: = " is the equivalence operator, and assigns a value to a nonterminal.

(c) "|" is the alternative operator. It indicates that one or the other of the symbols it separates is to be included in the expansion of the nonterminal. For instance, <x> :: = <y> | <z> means that <x> is either <y> or <z>.

(d) "[ ]" encloses one or more optional symbols. For instance, <x> :: = [<y>] <z> means that <x> is either <y> <z> or <z>.

(e) "( )" encloses a group of symbols.

(f) "<x>*" denotes the possible repetition of the symbol <x>, one or more times. For instance, <x> :: = .<y>* could also be expressed by the recursive rule <x> :: = <y> | <x> <y>.

(g) "l{ }" denotes an ordered list of elements.

(h) "p{ }" denotes an unordered set formed with name-value pairs, or a property list. A name-value pair consists of two symbols. The first symbol must be a nonterminal and the second can be any symbol.

Specification

```
<document body>::= <PD>

<PD>::= <PE> | <seq-pair> | <sim-pair> | <ind-pair> | <doc-pair>

<seq-pair>::= p{ SEQUENTIAL l{ <PD>* } }

<sim-pair>::= p{ SIMULTANEOUS l{ <PD>* } }

<ind-pair>::= p{ INDEPENDENT l{ <PD>* } }

<doc-pair>::= p{ DOC <document body> }

<PE>::= <tx-pair> | <vx-pair> | <ix-pair> | <gx-pair> | <fx-pair> | <ss-pair>

<tx-pair>::= p{ TEXT <text structure> }

<vx-pair>::= p{ VOICE <voice structure> }

<ix-pair>::= p{ IMAGE <image structure> }

<gx-pair>::= p{ GRAPHICS <graphics structure> }

<fx-pair>::= p{ FACSIMILE <facsimile structure> }

<ss-pair>::= p{ SPREADSHEET <spreadsheet structure> }

<version number>::= ; index of the version of the protocol

<spatial control>::= p{ LEFT <integer>
                        TOP <integer>
                        WIDTH <integer>
                        HEIGHT <integer> }

<tx-structure>::= p{ PROTOCOL <text protocol>
                     VERSION <version number>
                     [ SPATIAL <spacial control> ]
                     [ PARAMETERS <text parameters> ]
                     DATA <text medium>}

<text protocol>::= PARAGRAPH | ITEMIZE | ENUMERATE | VERBATIM

<text parameters>::= p{ [ CAPTION <caption string> ]
                        [ FONT <font name> ]
                        [ SIZE <font size> ]
                        [ STYLE <font style> ] }

<text medium>::= l{ <text string datum>* }

<vx-structure>::= p{ PROTOCOL <voice protocol>
                     VERSION <version number>
                     [ SPATIAL <spacial control> ]
                     [ PARAMETERS <voice parameters> ]
                     DATA <voice medium>}

<voice protocol>::= LPCM | PCM

<voice parameters>::= p{ [ CAPTION <caption string> ] }

<voice medium>::= l{ <bit string datum>* }
```

```
<image structure>::= p{   PROTOCOL   <image protocol>
                          VERSION    <version number>
                          [SPATIAL   <spatial control>]
                          PARAMETER  <image params>
                          DATA       <image medium>      }

<image protocol>::= BITMAP | GREYSCALE | RGB

<image params>::=   p{    [CAPTION   <caption string>]
                          HEIGHT     <# of lines in image>
                          [BITS/PIXEL <# of bits per pixel>]
                          [TABLE     l{ <color specifiers>*}]
                          WIDTH      <# pixels per line>    }

<image medium>::=   l{    <bit string datum>*    }
```

References

[1] Postel J., C. Sunshine, and D. Cohen, "The ARPA Internet Protocol", Computer Networks,Vol. 5, No. 4, July 1981.

[2] Postel, J., "Internet Protocol", RFC-791, USC/Information Sciences Institute, September 1981.

[3] DoD Military Standard, "Internet Protocol", MIL-STD-1777, Department of Defense, August 1983.

[4] Postel, J., "Transmission Control Protocol", RFC-793, USC/Information Sciences Institute, September 1981.

[5] DoD Military Standard, "Transmission Control Protocol", MIL-STD-1778, Department of Defense, August 1983.

[6] Sunshine, C., " Interconnection of Computer Networks", Computer Networks, Vol. 1, No. 3, January 1977.

[7] Postel, J., "User Datagram Protocol", RFC-768, USC/Information Sciences Institute, August 1980.

[8] Postel J., and J. Reynolds, "File Transfer Protocol (FTP)", RFC-959, USC/Information Sciences Institute, October 1985.

[9] Postel J., "An Internetwork Message Structure", Sixth Data Communications Symposium, ACM/IEEE, November 1979.

[10] Postel, J., "Internet Message Protocol", RFC-759, USC/Information Sciences Institute, August 1980.

[11] Postel, J., "Internet Multimedia Mail Transfer Protocol", RFC-759-revised, MMM-11, USC/Information Sciences Institute, March 1982.

[12] Reynolds, J., et al., "The DARPA Experimental Multimedia Mail System", IEEE Computer,Vol. 18, No. 10, October 1985.

[13] Finn G., and J. Postel, "Data Structures and Presentation Control for Multimedia Computer Mail", Proceedings of EASTCON, November 1981.

[14] Postel, J., "A Structured Format for Transmission of Multimedia Documents", RFC-767, USC/Information Sciences Institute, August 1980.

[15] Postel, J., "Internet Multimedia Mail Document Format", RFC-767-revised, MMM-12, USC/Information Sciences Institute, March 1982.

[16] Leiner B., R. Cole, J. Postel, and D. Mills, "The DARPA Internet Protocol Suite", IEEE INFOCOM85, Washington D.C., March 1985. Also in IEEE Communications Magazine, Vol. 23, No. 3, March 1985, and as ISI-RS-85-153.

[17] Postel J., "Internetwork Applications Using the DARPA Protocol Suite", IEEE INFOCOM85, Washington D.C., March 1985. Also as ISI-RS-85-151.

[18] Feinler, J., "Protocol Handbook", DDN Network Information Center, SRI International, December 1985.

[19] Sollins, K., "The TFTP Protocol", RFC-783, MIT Laboratory for Computer Science, June 1981.

[20] Katz, A., "An Experimental Internetwork Multimedia Mail System", Proceedings IFIP 6.5 Working Conference On Computer Message Services, Nottingham, England, May 1984. Also available as ISI-RS-84-134, USC/Information Sciences Institute, June 1984.

[21] Thomas, R., et al., "Diamond: A Multimedia Message System Built on a Distributed Architecture", IEEE Computer,Vol. 18, No. 12, December 1985.

[22] International Organization for Standardization, "Writing Paper and Certain Classes of Printed Matter - Trimmed Sizes - A and B Series", Technical Report ISO 216-1975 (E), ISO, 1975.

[23] Finn, G., "A Format for the Construction of Multimedia Messages", MMM-27, USC/Information Sciences Institute, September 1985.

[24] Garcia-Luna-Aceves, J., A. Poggio, and D. Elliott, "Research into Multimedia Message System Architecture, Final Report", Contract No. N00039-80-C-0658, SRI Project 5363, SRI International, February 1984.

[25] Katz, A., "Format for Bitmap Files", RFC-797, USC/Information Sciences Institute, September 1981.

[26] Postel, J., "Rapicom-450 Facsimile File Format", RFC-769, USC/Information Sciences Institute, September 1980.

[27] Cohen, D., "A Network Voice Protocol (NVP-II)", USC/Information Sciences Institute, April 1981.

[28] Travers, V., "The Representation of Graphics Objects in Multimedia Messages", Bolt Beranek and Newman, Report No. DPC-39, June 1983.

[29] Travers, V., "Syntax and Semantics of Graphics in Multimedia Messages", Bolt Beranek and Newman, Report No. DPC-40, June 1983.

[30] Crowley, T., "A Machine-Independent Standard for Stored Spreadsheets", Bolt Beranek and Newman, April 1983.

[31] Katz, A., "MMM - A Multimedia Mail System User Interface", USC/Information Sciences Institute, July 1983.

[32] Katz A., "Decoding Facsimile Data from the Rapicom-450", RFC-798, USC/Information Sciences Institute, September 1981.

[33] Agarwal, A., M. O'Connor, and D. Mills, "Dacom 450/500 Facsimile Data Transcoding", RFC-803, COMSAT, November 1981.

[34] DeSchon, A., "Facsimile Support Programs User Guide", USC/Information Sciences Institute, September 1983.

[35] Reynolds, J., "How to Create and Print a Bitmap Using a Hardcopy Picture", USC/Information Sciences Institute, August 1983.

[36] CCITT, "Recommendation T.4 -- Standardization of Group 3 Facsimile Apparatus for Document Transmission". Also as RFC-804, January 1982.

[37] Hoffstetter, E., J. Tierney, and O. Wheeler, "Microprocessor Realization of a Linear Predictive Vocoder", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-25, No. 5, October 1977.

[38] O'Leary, G., "Local Access Area Facilities for Packet Voice", International Conference on Computer Communication, Atlanta, Georgia, October 1980..

[39] Malpass, M., and J. Feldma, "A User's Guide for the Lincoln LPC Speech Processing Peripheral", PSST-2, MIT Lincoln Laboratory Project Report, April 1983.

[40] Cole, R., "Dialing in the WB Network", W-Note 26, USC/Information Sciences Institute, April 1981.

[41] Forgie, J., "ST - A Proposed Internet Stream Protocol", IEN-119, MIT Lincoln Laboratory, September 1979.

[42] Poggio, A., et al., "CCWS: A Computer-Based Multimedia Information System", IEEE Computer,Vol. 18, No. 10, October 1985.

Note: In the above references, the term "RFC" refers to papers in the DARPA "Request for Comments" series and "IEN" refers to DARPA "Internet Experiment Notes". These notes may be obtained from the DDN Network Information Center, SRI International, Menlo Park, California, 90425.

END

12-86

DTIC